Facultad de Ciencias Astronómicas y Geofísicas, UNLP, Argentina.
Universidad Nacional de Río Negro, Sede Andina, Argentina.
Instituto de Astrofísica de La Plata, UNLP–CONICET, Argentina.

La Plata
Variational Indicators code

A program to compute a suite of
variational chaos indicators

User Guide

for Version 2.0.2 (CONTROL)

Authors
D. D. Carpintero, N. P. Maffione, L. A. Darriba

Last update: December 2020

Contents

Chapter 1

The `LP-VIcode` project

1.1 Motivation

The correct analysis of a given dynamical system rests on the reliable identification of the chaotic or regular behaviour of its orbits. The most commonly used tools for such analyses are based either on the study of the fundamental frequencies of the trajectories, or on the study of the evolution of the deviation vectors, the so–called variational chaos indicators (CIs hereafter). Therefore, it seems very useful to have a tool with which one can compute several CIs in an easy and fast way. This is the main motivation of the `LP-VIcode` (the acronym for *La Plata–Variational Indicators code*).

The alpha version of the code was first introduced in [7]. The first stable version of the code was version 1.0.2, codename: `KAOS` and introduced in [3]. Hereinafter we present the last and current stable version of the code, version 2.0.2, codename: `CONTROL`.

1.2 Brief description of the `LP-VIcode`

The `LP-VIcode` is a program to compute a suite of CIs.

The library of CIs in the present version include the following: a) the Lyapunov Indicators, also known as Lyapunov Characteristic Exponents, Lyapunov Characteristic Numbers or Finite Time Lyapunov Characteristic Numbers (LIs, [1, 2]); b) the Mean Exponential Growth factor of Nearby Orbits (MEGNO, [4, 5]); c) the Slope Estimation of the largest Lyapunov Characteristic Exponent (SElLCE, [5]); d) the Smaller ALignment Index (SALI, [19]); e) the Generalized ALignment Index (GALI, [20, 21]); f) the Fast Lyapunov Indicator (FLI, [10, 11]); g) the Orthogonal Fast Lyapunov Indicator (OFLI, [9]); h) the Spectral Distance (SD, [23]); i) the dynamical Spectra of Stretching Numbers (SSNs, [22, 6]); and j) the Relative Lyapunov Indicator (RLI, [17, 18])[1]. The main achievement of the code is its speed: neither the orbit nor any of the sets of variational equations are computed more than once in each time step, even when they may be requested by more than one CI.

[1]A minimal package of well behaved CIs (called CIsF for *Chaos Indicators Function*) for analysing a general Hamiltonian system is studied in [12, 8, 13].

The program is written in standard `Fortran77`[2], except for the common nonstandard extensions `DO-ENDDO`, `INCLUDE`, `DOWHILE`, lowercase characters, inline comments, names longer than 6 characters, and names containing the nonstandard character "_". All real variables are `DOUBLE PRECISION`.

The program reads initial conditions (hereafter i.c.) for one or more orbits, and integrates them, computing at the same time the set of CIs chosen by the user. The integrator is a Bulirsch–Stoer routine.

1.3 Future versions

The main goal of the `LP-VIcode` project is to cluster in a single, easy-to-use tool the plethora of CIs that are most widely used being the starting point the `LP-VIcode KAOS`. We intend to motivate researchers to collaborate with their own methods in developing newer versions of the code with larger CIs' libraries and in order to achieve this goal it is critical to offer a very versatile tool. Therefore, we welcome people who might be interested in making the code more *user friendly.* For example, changing the present command–driven interface to a menu–driven interface, etc.

Furthermore, the next step in the `LP-VIcode` family is to add `SMART`: an automatic differentiation module as a preprocessing step. Stay tune to get `SMART`!

[2]There is also a projected version written in `Fortran90` for parallel computing, though in an early developing stage.

Chapter 2

Basics of `LP-VIcode`

2.1 Getting started

In order to use `LP-VIcode` the user needs:

1. a text editor to edit the files: `*.pav, LP-VIcode.in` and `LP-VIcode.par`;
2. a Fortran compiler to compile and execute the source Fortran code: `LP-VIcode.for` and `LP-VIcode.par`.

The main files of `LP-VIcode` (`LP-VIcode.for`, which is fully commented for easier understanding, and its companion `LP-VIcode.par`), an input data file (`LP-VIcode.in`), files defining the 2D Hénon–Heiles model, the 2D logarithmic potential and a triaxial extension of the well-known Navarro-Frenk-White profile [24] as sample test potentials (`henon-heiles.pav`, `logarithmic2.pav`, `nfwtriaxial.pav`, respectively) and examples of input files of i.c. for each potential (`in-hh.dat`, `in-log.dat` and `in-nfwtriaxial.dat`, respectively)[1], can be found at the url:

http://www.fcaglp.unlp.edu.ar/LP-VIcode/

After giving all the input data (see Section 2.2), the user should compile the `LP-VIcode.for` program. For a Unix-based system, this typically requires a command like

```
mymachine> gfortran -o your_executable LP-VIcode.for
```

which yields an executable file named `your_executable`. To run this executable, the user typically types

```
mymachine> ./your_executable
```

In order to show how to use `LP-VIcode` we will apply below the code to a simple 2–dimensional logarithmic potential (hereafter 2dLP), describing every step. Text referring to this particular example will be written in violet.

[1] In Appendix A.2 we apply `LP-VIcode` to the 2D Hénon–Heiles potential and the 3D triaxial NFW profile using the abovementioned input files.

2.2 INPUT data

I) The user must provide the potential in which the orbits have to be integrated, by means of a file written in `Fortran77` with the file extension: `.pav` (for the 2dLP, we named the file `logarithmic2.pav`). This `.pav` file has to be included at the end of `LP-VIcode.for` and should contain the following:

1. `NEW FOR VERSION 2.0.2` A subroutine called `potinit` where the user sets the dimension of the potential and initializes its parameters in any desired way. The dimension of the potential should be included in a `COMMON` named `dimpot`, whereas the parameters should be included in another named `COMMON` (any name) shared with the routines that compute the potential, the acceleration and the variational equations (see below).[2]

   Below, the first section of the `logarithmic2.pav` file where the subroutine `potinit` is placed. In this example, the dimension and the parameters are given their values by means of simple assignments, though they may be read from a file or assigned in any other way.

   SUBROUTINE potinit in logarithmic2.pav

```
*************************************************************
* Logarithmic 2D potential
* Parameters: q: flattening (ratio of semiaxes)
*             rc: core radius
*             v02: normalization constant
*************************************************************
      SUBROUTINE potinit
* Initialize dimension and parameters of the potential
*************************************************************
      INTEGER dim
      COMMON /dimpot/dim

      DOUBLE PRECISION q,rc,v02
      COMMON /param/q,rc,v02

      dim=2
      q=0.7d0
      rc=0d0
      v02=1d0
      END
```

2. A `FUNCTION pot(t,x,n)` in which the potential is computed from the time `t` and the `n` coordinates `x` of the phase space. The parameters should be received through the

[2]In case of using the NEMO front-end for the `LP-VIcode`, see further details at `https://github.com/teuben/nemo/blob/master/man/man1/runlpvi.1`.

`COMMON` described in `potinit`. For the 2dLP, the potential Φ is:

$$\Phi(x,y) = \frac{1}{2} v_0^2 \ln \left[x^2 + \left(\frac{y}{q}\right)^2 + r_c^2 \right].$$

FUNCTION pot in logarithmic2.pav

```
*************************************************************
      FUNCTION pot(t,x,n)
      INTEGER n
      DOUBLE PRECISION pot,t,x(n)
* t = time
* x(n) = coordinates of the phase space
* n = dimension of the phase space
*************************************************************
      DOUBLE PRECISION q,rc,v02
      COMMON /param/q,rc,v02

      DOUBLE PRECISION arg

      arg=x(1)**2+(x(2)/q)**2+rc**2
      IF(arg.GT.0d0)THEN
         pot=0.5d0*v02*LOG(arg)
      ELSE
         pot=-1d38
      ENDIF
      END
```

3. A `SUBROUTINE acelera(t,x,n,acc)` in which the accelerations ($\ddot{x}$ and $\ddot{y}$) are computed from the same data as the potential above. For the 2dLP, the accelerations are:

$$\begin{cases} \ddot{x} = -\dfrac{\partial \Phi}{\partial x} = -\dfrac{v_0^2}{x^2 + y^2/q^2 + r_c^2} x\,; \\[2ex] \ddot{y} = -\dfrac{\partial \Phi}{\partial y} = -\dfrac{v_0^2}{x^2 + y^2/q^2 + r_c^2} \dfrac{y}{q^2}\,. \end{cases}$$

SUBROUTINE acelera in logarithmic2.pav

```
*************************************************************
      SUBROUTINE acelera(t,x,n,acc)
      INTEGER n
      DOUBLE PRECISION t,x(n),acc(n/2)
* t = time
* x(n) = coordinates of the phase space
* n = dimension of the phase space
* acc(n/2) = accelerations (acc(1) = d[vx]/dt, etc)
*************************************************************
      DOUBLE PRECISION q,rc,v02
      COMMON /param/q,rc,v02

      DOUBLE PRECISION aux

      aux=v02/(x(1)**2+(x(2)/q)**2+rc**2)
      acc(1)=-x(1)*aux
      acc(2)=-x(2)*aux/q**2
      END
```

4. A `SUBROUTINE variac(t,x,dx,n,dax)` in which the variational equations corresponding to the velocities (only) are computed from the same data as the potential above, plus the deltas of the phase space. The variational equations corresponding to the velocities for the 2dLP are:

$$\begin{cases} (\delta\dot{x}) & = & \left[-\frac{v_0^2}{x^2+y^2/q^2+r_c^2} + \frac{2v_0^2x^2}{(x^2+y^2/q^2+r_c^2)^2}\right]\delta x + \left[\frac{2v_0^2xy/q^2}{(x^2+y^2/q^2+r_c^2)^2}\right]\delta y\,; \\ (\delta\dot{y}) & = & \left[\frac{2v_0^2xy/q^2}{(x^2+y^2/q^2+r_c^2)^2}\right]\delta x + \left[-\frac{v_0^2/q^2}{x^2+y^2/q^2+r_c^2} + \frac{2v_0^2y^2/q^4}{(x^2+y^2/q^2+r_c^2)^2}\right]\delta y\,. \end{cases}$$

SURBOUTINE variac in logarithmic2.pav

```
*************************************************************
      SUBROUTINE variac(t,x,dx,n,dax)
      INTEGER n
      DOUBLE PRECISION t,x(n),dx(n),dax(n/2)
* t = time
* x(n) = coordinates of the phase space
* dx(n) = differentials (deltas) of the coordinates of the phase space
* n = dimension of the phase space
* dax(n/2) = derivatives (dax(1) = d[dvx]/dt, etc)
*************************************************************
      DOUBLE PRECISION q,rc,v02
      COMMON /param/q,rc,v02

      LOGICAL prim
      DOUBLE PRECISION arg,par,q2,rc2,arg2,aux1,aux2
      SAVE prim,q2,rc2
      DATA prim/.TRUE./

      IF(prim)THEN
         prim=.FALSE.
         q2=q**2
         rc2=rc**2
      ENDIF
      arg=x(1)**2+x(2)**2/q2+rc2
      arg2=arg**2
      par=x(1)*dx(1)+x(2)*dx(2)/q2
      aux1=2d0*v02*par/arg2
      aux2=v02/arg
      dax(1)= aux1*x(1)-aux2*dx(1)
      dax(2)=(aux1*x(2)-aux2*dx(2))/q2
      END
```

Once the file with these routines is ready[3], its name should be `INCLUDE`'d at the end of the file `LP-VIcode.for`, replacing any other `*.pav` file which happens to be mentioned there. See below the last lines in the `LP-VIcode.for` for the 2dLP potential.

Last section of the LP-VIcode.for (for 2dLP potential)

```
*************************************************************
* Potential, acceleration, variational equations (user-provided)
* 2D potentials
c      INCLUDE 'henon-heiles.pav'
      INCLUDE 'logarithmic2.pav'
* 3D potentials
c      INCLUDE 'nfwtriaxial.pav'
*************************************************************
```

II) The user must also provide the input parameters requested in the file `LP-VIcode.in`. These data are, one per record of the file `LP-VIcode.in`:

1. The name of the input file where the i.c. of the orbits are to be found. In this last file, if the potential is of dimension `n`, the i.c. are given as `n` coordinates plus `n` velocities, in that order, one orbit per record. For example, suppose the i.c. for our example potential 2dLP are to be written in the file `in-log.dat`. The initial condition $(x, y, \dot{x}, \dot{y}) = (0.2, 0.34, 0.5, 0.1235)$ should be written in that file in the following way (or any other equivalent legal Fortran77 way):

   in-log.dat

   ```
   0.2d0   0.34d0   0.5d0   0.1235d0
   ```

2. Prefix of the output files. The output is dumped in files with this prefix (in the case of the 2dLP, we chose the prefix `log`) appended with the following suffixes:

[3]With the `SMART` module the subroutines `acelera` and `variac` will be computed automatically as a preprocessing step, without any further effort from the user.

- `.ene` for the energy conservation after the integration;
- `.orb` for the orbit;
- `.geo` for physical and geometrical properties of the orbit;
- `.lyap` for the LIs;
- `.sali` for the SALI;
- `.gali` for the GALIs;
- `.sd` for the SD;
- `.ssn` for the SSNs used to compute the SD;
- `.rli` for the RLI and the maximal LI computed as the rate of separation of two infinitesimally separated orbits;
- `.megno` for the MEGNO and the SElLCE;
- `.fli` for the FLI and the OFLI;

If there are more than one orbit to integrate, they are separated in these files with blank lines.

3. Step of integration for the orbit and the variational equations.

4. `NEW FOR VERSION 2.0.2` Initial and final time of integration for the orbit (the variational equations may end sooner than this if the corresponding indicator saturates[4]).

First part of LP-VIcode.in (example for the 2dLP potential)

```
# LP-VIcode version 2.0.2
# Initial conditions file (max. 50 characters)
in-log.dat
# Prefix for output files (max. 50 characters)
log
# Step of integration
0.05d0
# Initial time; final time of integration
0.d0 25000.d0
```

The user may also wish to integrate each orbit of an ensemble from different initial times and/or until different final times. To this end, he/she must replace the initial and/or final time with a "`f`" (a shortcut for "file") in the file `LP-VIcode.in`; this letter tells the code to look after the times of integration in the input file, after the initial conditions of each orbit (one number for each "`f`"). In the case of our example and in order to start at `0.d0` and also ends by `25000.d0`, the file `in-log.dat` should be:

in-log.dat

```
0.2d0   0.34d0   0.5d0   0.1235d0   0.d0 25000.d0
```

[4]The CIs that have a threshold value, and therefore may saturate, are the following: SALI, GALIs, MEGNO, FLI and OFLI.

while in the input file, it must reads:

LP-VIcode.in (example for the 2dLP potential)

```
# Initial time; final time of integration
f f
```

5. Screen and orbit flags. The first flag (one digit) controls whether the progress of the computation, as well as the energy conservation at the end of the integration of each orbit (`DE`), should be output to screen (flag = 1) or not (flag = 0)[5]. The second one controls whether the orbit(s):

   - 0 = should not be output to a file;
   - 1 = or should;
   - `NEW FOR VERSION 2.0.2` 2 = or not only should be output to a file but also with a companion archive of extension `.geo` where the physical and geometrical properties of that(those) orbit(s) (the angular momentum, the radius of the last apocenter and pericenter and the number of periods since the first apocenter) are also dumped.

   `WARNING: CONFIGURATION INCOMPATIBILITY.` In the latter case (flag = 2), `ndim` should be 2 or 3, i.e. only two- and three–dimensional potentials are admited. Otherwise, the corresponding output is automatically cancelled and the message:

   `'# degrees of freedom:  output disabled.'`

   appears in the `.geo` file.

   Example of screen output

   ```
   Orbit #     1 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% DE = 4.12E-14
   Orbit #     2 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% DE = 1.71E-13
   Orbit #     3 10% 20% 30% 40% 50%
   ```

6. CIs flag vector. A flag 7–vector is used to control whether to compute/output the CIs, in this order: LIs, SALI, GALIs, SD & SSNs, RLI & LImax, MEGNO & SElLCE, FLI & OFLI. Each flag may take one of the following values:

   - 0 = do not compute the corresponding CI, therefore also do not output;
   - 1 = compute the CI, and output along the integration for the whole integration interval or until the CI saturates;
   - 2 = compute and output only the final values. If this option is chosen, the final time (which corresponds to the saturation time if the CI saturated) and the initial conditions are also output to the file.

[5]The index of the orbit can go as far as 99999, beyond that number the index exceeds the available `FORMAT` and there is no further registration of the latter.

For the 2dLP we are interested in computing the time evolution of all the CIs, dumping the values of the orbit as well as the additionals in a file, and watching the progress of the computation. Then, the screen and orbit flags are set to 1 and 2, respectively and the other 7 flags corresponding to the CIs flag vector are all set to 1 in `LP-VIcode.in`.

7. Number of steps between outputs: in case any of the abovementioned components of the flag 7–vector, or the flag associated with the orbit, is set to 1 or 2, this parameter indicates how many steps have to be integrated before an output is dumped to the files. This holds for every output file except for the `.ssn` file (see below Table 2.1 for further details), and in case of the `.orb` file, the first line corresponds to the initial condition, i.e. $t =$ initial time.

8. Initial deviation vectors: an integer indicates whether the initial deviation vectors should be generated at random (= `0`), or should be generated at random and also be orthonormalized (= `1`) or their values are to be provided by the user (= `2`). In this last case, the user should edit subroutine `condini` in the source file `LP-VIcode.for` and write down there `2n` deviation vectors of `2n` components each (being `n` the dimension of the potential). The source text provides more detailed instructions as comments. In the case of the 2dLP we chose random orthonormal initial deviation vectors, so we set this flag to 1.

9. `NEW FOR VERSION 2.0.2` SALI reinitialization flag: in case the user is dealing with a time-dependent potential, the SALI can be used to determine the windows of regular/chaotic behaviour of the orbits (see, for instance, [14, 15, 16]). If the flag is not activated (flag = `0`), then the computation continues with a normal saturation strategy. In case the flag is activated (flag = `1`), the deviation vectors are restarted when the saturation is reached.

   `WARNING: CONFIGURATION INCOMPATIBILITY.` This election forces the program to output the SALI at every writing step, and cancel any output of the GALIs, SD, SSNs, RLI or LImax (they share their deviation vectors with the SALI). Also, the generation of the initial deviation vectors are changed to be at random and the following warning sign is deployed on the screen to alert the user of the forced changes:

   ```
   'Warning:  Restarting SALI is only compatible with SALI output for all
   t, with no computing GALIs, SD/SSNs or RLI/LImax, and with random
   initial deviation vectors.
   Input parameters modified to satisfy these conditions'
   ```

Second part of LP-VIcode.in (example for the 2dLP potential)

```
# Screen (0=no, 1=yes) & orbit (0=no output, 1=output, 2=output additionals)
1 2
# Indicators: 0=don't compute, 1=output for all t, 2=output only last value
# LIs, SALI, GALIs, SD & SSNs, RLI & LImax, MEGNO & SElLCE, FLI & OFLI
1 1 1 1 1 1 1
# Nr. of steps between outputs (when orbit or indicators are = 1)
200
```

```
# Initial dev. vectors (0 = at random, 1 = random orthonormal, 2=fixed)
1
# SALI & potential(t): normal saturation (=0) or restart dev. vectors (=1)
0
```

III) Fixed parameters of the program, file `LP-VIcode.par`:

In this file there is the parameter `ndim` that should be set to the dimension of the potential being used.
`WARNING: CONFIGURATION INCOMPATIBILITY.` This parameter is used to define the dimension of many arrays, and is checked against the dimension of the potential declared in the corresponding file `.pav` to avoid forgetting to change it when the potential is changed to another one with another dimension. If this happens, the following error message appears:

```
'Dim of potential in *.pav <> dim in LP-VIcode.par'
```

The rest of the parameters of this file are not meant to be changed, but they certainly may be modified if the user wishes. They are all described in the file `LP-VIcode.par` itself.

LP-VIcode.par

```
*************************************************************
* LP-VIcode.par: parameters of LP-VIcode.for (Version 2.0.2)
*************************************************************
* ndim: dimension of the potential.
      INTEGER ndim
      PARAMETER (ndim=2)
*************************************************************
* Dimension of the phase space
      INTEGER ndimf
      PARAMETER (ndimf=2*ndim)
* Maximum nr. of steps to be integrated
      INTEGER npmax
      PARAMETER (npmax=1000000)
* Nr. of indicators of chaos
      INTEGER nind
      PARAMETER (nind=7)
* Initial separation of orbits (RLI)
      DOUBLE PRECISION rlieps
      PARAMETER (rlieps=1d-12)
* Number of slots in the histogram (spectral distance)
      INTEGER nbin
      PARAMETER (nbin=500)
* Format for the output of numbers
      CHARACTER fo*11
      PARAMETER (fo='(e22.16,1x)')
* Saturation values of the indicators:
```

```
* SALI, GALI
      DOUBLE PRECISION satsal
      PARAMETER (satsal=1d-16)
* FLI, OFLI
      DOUBLE PRECISION satfli
      PARAMETER (satfli=1d16)
* MEGNO
      DOUBLE PRECISION satmeg
      PARAMETER (satmeg=30d0)
```

The parameter `rlieps` affects only the first component of the initial deviation vector associated with the second orbit used to compute the RLI. Should the user want to change more components of that vector (or another one), he/she must change the corresponding command line in subroutine `condini` (`I/O and initialization module` in `LP-VIcode.for`).

2.3 OUTPUT data

Table 2.1 shows the information provided in the output files in the case of dumping the time evolution of the CIs; the names of the files and the number of columns correspond to the example of the 2dLP. When this kind of output is chosen, different orbits are separated by blank lines.

In case of dumping just the final values, the information in the output files is like that presented in Table 2.2 for the example of the 2dLP, where $t_{\rm sat}^{\rm CI}$ indicates the CI's time of saturation. If the CI does not reach the saturarion value, $t_{\rm sat}^{\rm CI}$ indicates the total integration time.

2.3.1 Strategy for runtime errors

If there is a runtime error, the computation is not stopped. However, in order to alert the user, the code prints an error message both in the file with extension `ene` and on the screen (if the corresponding flag is switched on). If the time evolution of the CIs are being dumped to file, a blank line is added at that instant, before the next orbit starts. If only the final values are being dumped to file, the CI's for the present orbit are all set to zero.

File	Number of columns	Values dumped in the columns
`log.ene`	3	# of orbit; energy, energy conservation
`log.orb`	$1 + 2n = 5$	t; x, y, $\dot{x}$, $\dot{y}$
`log.geo`	8	t; final energy, $L_x, L_y, L_z, r_\mathrm{a}, r_\mathrm{p}$, # of periods
`log.lyap`	$1 + 2n = 5$	t; L1, L2, L3, L4
`log.sali`	2	t; SALI
`log.gali`	$1 + (2n - 1) = 4$	t; GALI_2, GALI_3, GALI_4
`log.sd`	2	t; SD
`log.ssn`	3	cell; SSN for 1st dev. vector, SSN for 2nd dev. vector
`log.rli`	3	t; RLI, max LI
`log.megno`	3	t; MEGNO, SElLCE
`log.fli`	3	t; FLI, OFLI

Table 2.1: Structure of the output files for the flag 7-vector: `1 1 1 1 1 1 1`. t is the time corresponding to each output value. "L_x", "L_y" and "L_z" are the "x", "y" and "z" components of the angular momentum, respectively. Also, "r_a" and "r_p" are the radius of the last apocenter and last pericenter, respectively, and "# of periods" is the number of periods since the first apocenter.

File	Number of columns	Values dumped in the columns
`log.ene`	3	# of orbit; energy, energy conservation
`log.lyap`	$2n + 2n = 8$	i.c.; L1, L2, L3, L4
`log.sali`	$2n + 2 = 6$	i.c.; SALI; $t_{\text{sat}}^{\text{SALI}}$
`log.gali`	$2n + (2n - 1) + (2n - 1) = 10$	i.c.; GALI$_2$, GALI$_3$, GALI$_4$; $t_{\text{sat}}^{\text{GALI}_2}$, $t_{\text{sat}}^{\text{GALI}_3}$, $t_{\text{sat}}^{\text{GALI}_4}$
`log.sd`	$2n + 1 = 5$	i.c.; SD
`log.rli`	$2n + 2 = 6$	i.c.; RLI, max LI
`log.megno`	$2n + 3 = 7$	i.c.; MEGNO, SElLCE; $t_{\text{sat}}^{\text{MEGNO}}$
`log.fli`	$2n + 4 = 8$	i.c.; FLI, OFLI; $t_{\text{sat}}^{\text{FLI}}$, $t_{\text{sat}}^{\text{OFLI}}$

Table 2.2: Structure of the output files for the flag 7-vector: `2 2 2 2 2 2 2`.

Chapter 3

Frequently asked questions

- About the project

  1. Can I compute indicators other than those based on the deviation vectors?

     No.

  2. Can I use it to study non–Hamiltonian systems?

     Of course. There is no Hamiltonian wired into the program; only equations of motion of a given dynamical system and its variational equations. The only restriction is that the accelerations should come from a potential.

  3. Where can I find information about the current status of the code as well as the downloadable files of the current version in order to use it?

     At the url:

     `http://www.fcaglp.unlp.edu.ar/LP-VIcode/`

  4. What should I do if I want my own personal CI to be part of the current official version of the code?

     Contact us and we will tell you how to proceed (contact information at the web page).

- About `LP-VIcode`

  1. Why does the code take less time to compute the CIs when compared to the computation of each CI separately?

     We have intertwined the equations so as to, for each time step, the equations of motion and the different sets of variational equations required for different CIs are integrated only once. For example: in computing the MEGNO and the FLI and the OFLI, only one variational equation is used for all of them. Moreover, if none of them are required, the corresponding variational equation is ignored. The same goes for the rest of CIs.

2. Why did we choose to write the code in `Fortran77`?

   When number-crunching is what is needed, there are two competing programming languages: `Fortran` and `C`. If one needs to fine tune the hardware, or use graphical interfaces, or send commands to the hardware, or use chunks of code written in low level languages, `C` is naturally the choice, since `Fortran` cannot handle those issues. However, if only numerical computation is all what is needed, the simplicity and speed of `Fortran` makes it the natural language to choose.

3. Is it not more convenient to save the output data in binary format?

   Sure! But binary files in `Fortran` are not portable. Thus, we chose to dump results in the less economical but portable ASCII format. Nevertheless, if you plan to work only on one machine, or on similar machines with identical compilers, you may easily change the `OPEN` and `WRITE` sentences of routine `output` to output in binary format, if that is what you want.

4. Is it easy to add my own personal indicator in the code?

   Not at all. The wiring of the equations makes the adding of a new indicator a delicate matter. However, if you fully understand what the routines `numbor`, `condini` and `saturation` do, you should have no problem in adding your favorite indicator to the suite.

5. Can I change the integrator routine?

   We have fine-tuned the Bulirsch-Stoer routine in order to optimise the computing time, by choosing different error thresholds for different indicators. So, it is no easy task to replace the integration routine already coded into `LP-VIcode`.

6. Is there a version for parallel computing?

   We are also coding our program in `Fortran90`. This version, however, is still being developed. We want it to be a true `Fortran90` code, i.e., not a mere translation of the `Fortran77` version, but a code which makes good use of the full power of `Fortran90`.

Acknowledgements

The authors wish to express their deep gratitude to Pablo Cincotta and Claudia Giordano for their fundamental contributions to the alpha version of the `LP-VIcode`. Also, their encouragement and participation in the project are very appreciated.

This work was supported with grants from the FCAG (UNLP), the IALP (UNLP-CONICET), the UNRN and the CONICET.

Bibliography

[1] Benettin, G., Galgani, L. & Strelcyn, J. 1976, Phys. Rev. A, 14, 2338-2345.

[2] Benettin, G., Galgani, L., Giorgilli, A. & Strelcyn, J. 1980, Meccanica, 15, Part I 9-20; Part II 21-30.

[3] Carpintero, D., Maffione, N. & Darriba, L. 2014, Astronomy and Computing, 5, 19-27.

[4] Cincotta, P. & Simó, C. 2000, A&A, 147, 205-228.

[5] Cincotta, P., Giordano, C. & Simó, C. 2003, Physica D, 182, 151-178.

[6] Contopoulos, G. & Voglis, N. 1996, CeMDA, 64, 1-20.

[7] Darriba, L., Maffione, N., Cincotta, P. & Giordano, C., 2012: 3rd La Plata International School on Astronomy and Geophysics "Chaos, Diffusion and Non-integrability in Hamiltonian Systems-Application to Astronomy". P.M. Cincotta, C.M. Giordano & C. Efthymiopoulos eds. Universidad Nacional de La Plata and Asociación Argentina de Astronomía Publishers, La Plata, Argentina, 345-366.

[8] Darriba, L., Maffione, N., Cincotta, P. & Giordano, C. 2012, IJBC, 22, 1230033 (33).

[9] Fouchard, M., Lega, E., Froeschlé, Ch. & Froeschlé, Cl 2002, CeMDA, 83, 205-222.

[10] Froeschlé, Cl., Gonczi, R. & Lega, E. 1997, Planet. Space Sci., 45, 881-886.

[11] Lega, E. & Froeschlé, Cl. 2001, CeMDA, 81, 129-147.

[12] Maffione, N., Darriba, L., Cincotta, P. & Giordano, C. 2011, CeMDA, 111, 285-307.

[13] Maffione, N., Darriba, L., Cincotta, P. & Giordano, C. 2013, MNRAS, 429, 2700-2717.

[14] Manos, T., Bountis, T. & Skokos, Ch. 2013, J. Phys. A: Math. Theor. 46 254017.

[15] Manos, T. & Machado, R.E.G. 2014, MNRAS, 438, 2201-2217.

[16] Machado, R.E.G. & Manos, T. 2016, MNRAS, 458, 3578-3591.

[17] Sándor, Z., Érdi, B. & Efthymiopoulos, C. 2000, CeMDA, 78, 113-123.

[18] Sándor, Z., Érdi, B., Széll, A. & Funk, B. 2004, CeMDA, 90, 127-138.

[19] Skokos, Ch. 2001, J. Phys. A: Math. Gen., 34, 10029-10043.

[20] Skokos, Ch., Bountis, T. & Antonopoulos, Ch. 2007, Physica D, 231, 30-54.

[21] Skokos, Ch., Bountis, T. & Antonopoulos, Ch. 2008, EPJST, 165, 5-14.

[22] Voglis, N. & Contopoulos, G. 1994, J. Phys. A: Math. Gen., 27, 4899-4909.

[23] Voglis, N., Contopoulos, G. & Efthymiopoulos, C. 1999, CeMDA, 73, 211, 220.

[24] Vogelsberger, M., White, S., Helmi, A., Springel, V. 2008, MNRAS, 385, 236-254.

Appendix A

Sample test potentials

Here we show complete examples of the `LP-VIcode` applied to two potentials. The first one is the well-known 2D Hénon–Heiles model:

$$\Phi_{\mathrm{H}} = \frac{1}{2}\left(x^2 + y^2 + 2x^2y - \frac{2}{3}y^3\right). \tag{A.1}$$

The second potential is a triaxial NFW profile [24]:

$$\Phi_{\mathrm{N}} = -\frac{A}{r_p}\ln\left(1 + \frac{r_p}{r_s}\right), \tag{A.2}$$

where A and r_s are constants, and

$$r_p = \frac{(r_s + r)r_e}{r_s + r_e}, \tag{A.3}$$

with

$$r_e = \sqrt{\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2}, \tag{A.4}$$

a, b, c constants, and $r = \sqrt{x^2 + y^2 + z^2}$. The values of the constants we used are listed in Table A.1.

A.1 The 2D Hénon–Heiles potential

Here we apply `LP-VIcode` to the 2D Hénon–Heiles model.

We want to compute the following CIs along the trajectory: the GALI$_3$, the RLI and the max LI, the MEGNO and the OFLI. This means that in the file `LP-VIcode.in` we need the flag 7–vector which refer to the CIs to be: `0 0 1 0 1 1 1` (with this choice for the flag vector, the code will also compute the GALI$_2$ and GALI$_4$, the SElLCE and the FLI). The values of the initial deviation vectors are taken random and orthonormal. Then the file `LP-VIcode.in` is:

A	4158670.1856267899
r_s	19.044494521343964
a	1.325882084000000
b	0.8626454020000000
c	0.7056058460000000

Table A.1: Constants used for the Φ_{N} potential.

LP-VIcode.in

```
# LP-VIcode version 2.0.2
# Initial conditions file (max. 50 characters)
in-hh.dat
# Prefix for output files (max. 50 characters)
hh
# Step of integration
0.05d0
# Initial time; final time of integration
0.d0 15000.d0
# Screen (0=no, 1=yes) & orbit (0=no output, 1=output, 2=output additionals)
1 0
# Indicators: 0=don't compute, 1=output for all t, 2=output only last value
# LIs, SALI, GALIs, SD & SSNs, RLI & LImax, MEGNO & SElLCE, FLI & OFLI
0 0 1 0 1 1 1
# Nr. of steps between outputs (when orbit or indicators are = 1)
100
# Initial dev. vectors (0 = at random, 1 = random orthonormal, 2=fixed)
1
# SALI & potential(t): normal saturation (=0) or restart dev. vectors (=1)
0
```

The value of the parameter `npmax` (in `LP-VIcode.par`) was set to 2 000 000.

The initial conditions are taken from [5] and they are included in the file `in-hh.dat`:

in-hh.dat

```
   0.d0  0.295456d0  0.407308431d0  0.d0
   0.d0  0.483d0     0.27898039d0   0.d0
```

```
0.d0  0.46912d0   0.291124891d0  0.d0
0.d0  0.509d0     0.254624859d0  0.d0
0.d0  0.56d0      0.164113781d0  0.112d0
```

The first orbit is close to a stable 1–periodic orbit at $(y, p_y) = (0.295456, 0)$ (we call it the *sp* orbit); the second orbit is like a stable quasi–periodic at $(y, p_y) = (0.483, 0)$ (*qp*), the third one at $(y, p_y) = (0.46912, 0)$ is also quasi–periodic but close to an unstable 4–periodic orbit (*up*); finally, the fourth and fifth orbits are irregular orbits, one inside a thick stochastic layer (*c1*) at $(y, p_y) = (0.509, 0)$, and the other one lying in a large chaotic sea (*c2*) at $(y, p_y) = (0.56, 0.112)$ (see [5] for details).

Once we have all the data for the input file `LP-VIcode.in` and `LP-VIcode.par`, we compile `LP-VIcode.for` and execute the file `LP-VIcode`. The total cpu time was about 30.137 s for a MacBook Air (11-inch, Early 2015): processor 1.6 GHz Dual-Core Intel Core i5; memory: 4 GB 1600 MHz DDR3, and running macOS Catalina 10.15.4 and `gfortran` compiler of `gcc` version 6.3.0, without any optimizations.

The flag for visual control of the processing was enabled, therefore on the screen should appear the progress of the computation and the energy conservation like we show below:

```
Control of the processing

Orbit #     1 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% DE = 3.22E-13
Orbit #     2 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% DE = 1.33E-12
Orbit #     3 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% DE = 3.08E-13
Orbit #     4 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% DE = 1.46E-12
Orbit #     5 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% DE = 1.98E-13
```

The output data is in the following files: `hh.ene`, `hh.gali`, `hh.rli`, `hh.megno` and `hh.fli`.

The file `hh.ene` shows the energy (`E`) and the conservation in the energy (`DE`) for the integration interval and for each of the five orbits:

```
hh.ene

1  E =    0.11799999984378129       DE =    3.2154128743047028E-013
2  E =    0.11800000000227606       DE =    1.3310915458119862E-012
3  E =    0.11799999987237160       DE =    3.0825154138663847E-013
4  E =    0.11799999974371828       DE =    1.4599903239015043E-012
5  E =    0.11799999989039131       DE =    1.9793488900209420E-013
```

The files `hh.gali`, `hh.rli`, `hh.megno` and `hh.fli` contain the data of the CIs. For example, the last two lines for the first orbit (i.e. *sp* orbit) of file `hh.fli` read:

```
hh.fli

0.1499500000000000E+05 0.5138152868435193E+03 0.1340629867756790E+01
0.1500000000000000E+05 0.5139749981345966E+03 0.1340629867756790E+01
```

On the other hand, the last two lines for the last orbit (i.e. *c2* orbit) of file `hh.fli` read:

```
hh.fli
0.1045000000000000E+04 0.5831029568285286E+16 0.5801158443157209E+16
0.1048950000000000E+04 0.1000000000000000E+17 0.1000000000000000E+17
```

which shows that the *sp* orbit (as well as the *qp* and *up* orbits) did not saturate before they reach the total integration time (i.e. 15000 u.t.), but the orbit *c2* (as well as the *c1* orbit) did.

Now we show in Figs. A.1 and A.2 the results for the five orbits and for the different CIs computed. It is clear that the behaviours of the CIs are the expected ones (see references).

This example might be useful to check the correct use of the code. To this end, a complete `pav` file of this potential is provided in the web page.

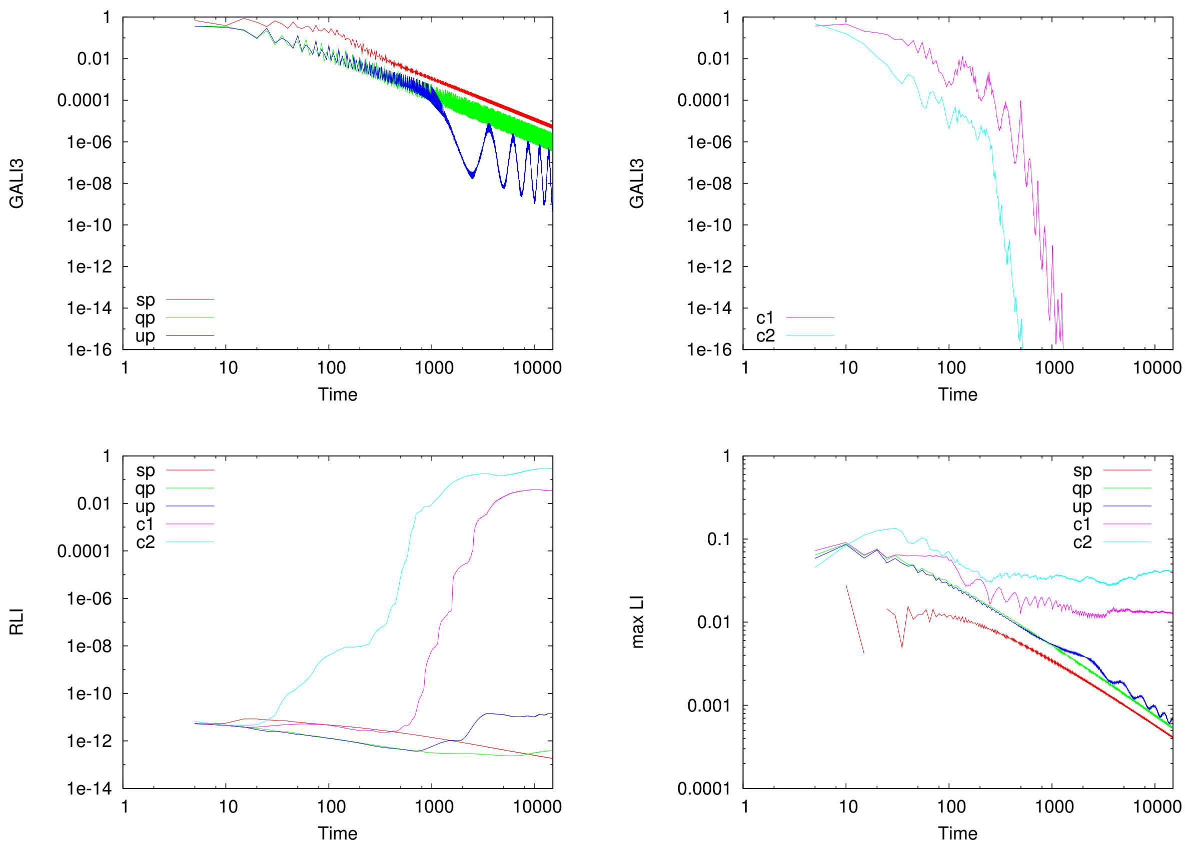

Figure A.1: Examples of the time evolution for the GALI$_3$ (top panels), the RLI (bottom left panel) and the max LI (bottom right panel) for 5 different orbits.

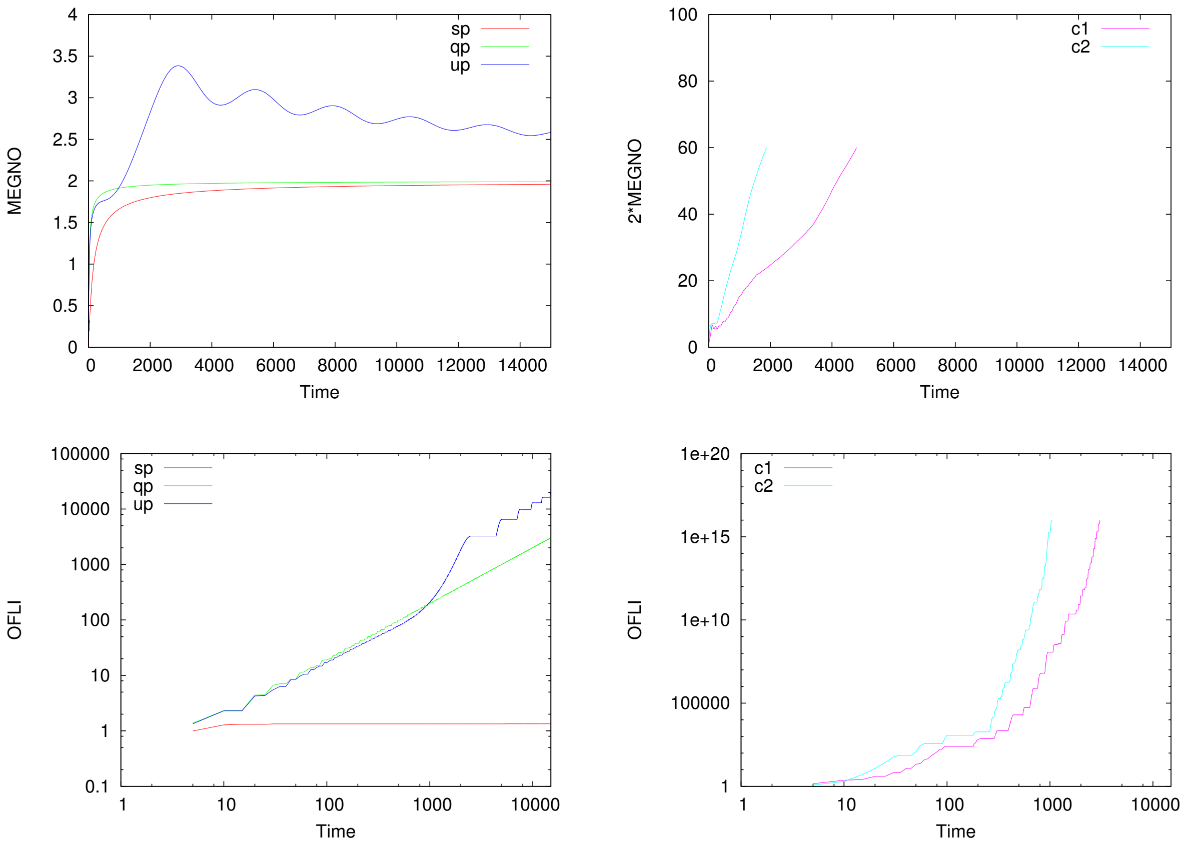

Figure A.2: Examples of the time evolution for the MEGNO (top panels) and the OFLI (bottom panels) for 5 different orbits.

A.2 The triaxial NFW profile

Here we apply `LP-VIcode` to the triaxial NFW profile described in Section A.

We want to compute the following CIs along the trajectory: the LIs, the SALI and the FLI. This means that in file `LP-VIcode.in` we need the flag 7–vector which refer to the CIs to be: `1 1 0 0 0 0 1` (with this choice the code will also compute the OFLI). The values of the initial deviation vectors are taken random and orthonormal. Since in this example we want the orbits to be integrated until different final times, we set the time of integration to zero, thus indicating that the individual times should be read from the file of initial conditions (see below). Then the file `LP-VIcode.in` is:

LP-VIcode.in

```
# LP-VIcode version 2.0.2
# Initial conditions file (max. 50 characters)
in-nfwtriaxial.dat
# Prefix for output files (max. 50 characters)
nfw
# Step of integration
0.0001d0
# Initial time; final time of integration
0.d0 0.d0
# Screen (0=no, 1=yes) & orbit (0=no output, 1=output, 2=output additionals)
1 0
# Indicators: 0=don't compute, 1=output for all t, 2=output only last value
# LIs, SALI, GALIs, SD & SSNs, RLI & LImax, MEGNO & SElLCE, FLI & OFLI
1 1 0 0 0 0 1
# Nr. of steps between outputs (when orbit or indicators are = 1)
100
# Initial dev. vectors (0 = at random, 1 = random orthonormal, 2=fixed)
1
# SALI & potential(t): normal saturation (=0) or restart dev. vectors (=1)
0
```

The value of the parameter `npmax` (in `LP-VIcode.par`) was chosen to be the standard: 1 000 000. However, the next error message appears:

```
'STOP lpvicode:  increase npmax in LP-VIcode.par'
```

Therefore, the value of the parameter `npmax` is 2 000 000.

The initial conditions are included in the file `in-nfwtriaxial.dat`. In each row, corresponding to an orbit, we add the desired time of integration after the values of the coordinates of the phase space, as shown below:

```
in-nfwtriaxial.dat
   7.004d0  1.566d0   1.757d0   90.786d0   65.561d0  -43.003d0  130.d0
   6.903d0  -1.000d0  1.020d0   -37.876d0  8.507d0   41.721d0  13.d0
   5.743d0  1.221d0   -0.576d0  -8.451d0   10.923d0  21.592d0  1.3d0
```

We call the orbits *(a)*, *(b)* and *(c)*, respectively.

Since the last example was a 2D potential, and now we are going to integrate a 3D potential, we changed the `dim` parameter in file `LP-VIcode.par` from 2 to 3, and changed the included `pav` file in file `LP-VIcode.for` to the corresponding new potential. After compiling and running, the total cpu time was 1 min 48 s.

The flag for visual control of the processing was enabled:

```
Control of the processing
 Orbit #      1 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% DE = 2.12E-13
 Orbit #      2 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% DE = 1.09E-14
 Orbit #      3 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% DE = 7.65E-15
```

The output data is in the following files: `nfw.ene`, `nfw.lyap`, `nfw.sali` and `nfw.fli`.

The file `nfw.ene` shows the energy (`E`) and the conservation in the energy (`DE`) for the integration interval and for each of the three orbits:

```
nfw.ene
 1  E =   -180966.55208172725       DE =    2.1244898347835213E-013
 2  E =   -188724.61491507938       DE =    1.0949138581408157E-014
 3  E =   -194085.76675494364       DE =    7.6476259857195925E-015
```

The files `nfw.lyap`, `nfw.sali` and `nfw.fli` contain the data of the CIs. For example, the last two lines for orbit *(a)* of file `nfw.sali` read:

```
nfw.sali
0.1299900000000000E+03 0.3255977012748573E-07
0.1300000000000000E+03 0.3376495601125773E-07
```

which shows that orbit *(a)*, as well as orbits *(b)* and *(c)*, did not saturate before they reached the total integration time (i.e. 130, 13 and 1.3 u.t., respectively).

We show in Fig. A.3 the results for the three orbits and for the different CIs computed.

This example might be also useful to check the correct use of the code. To this end, a complete `pav` file of this potential is provided in the web page.

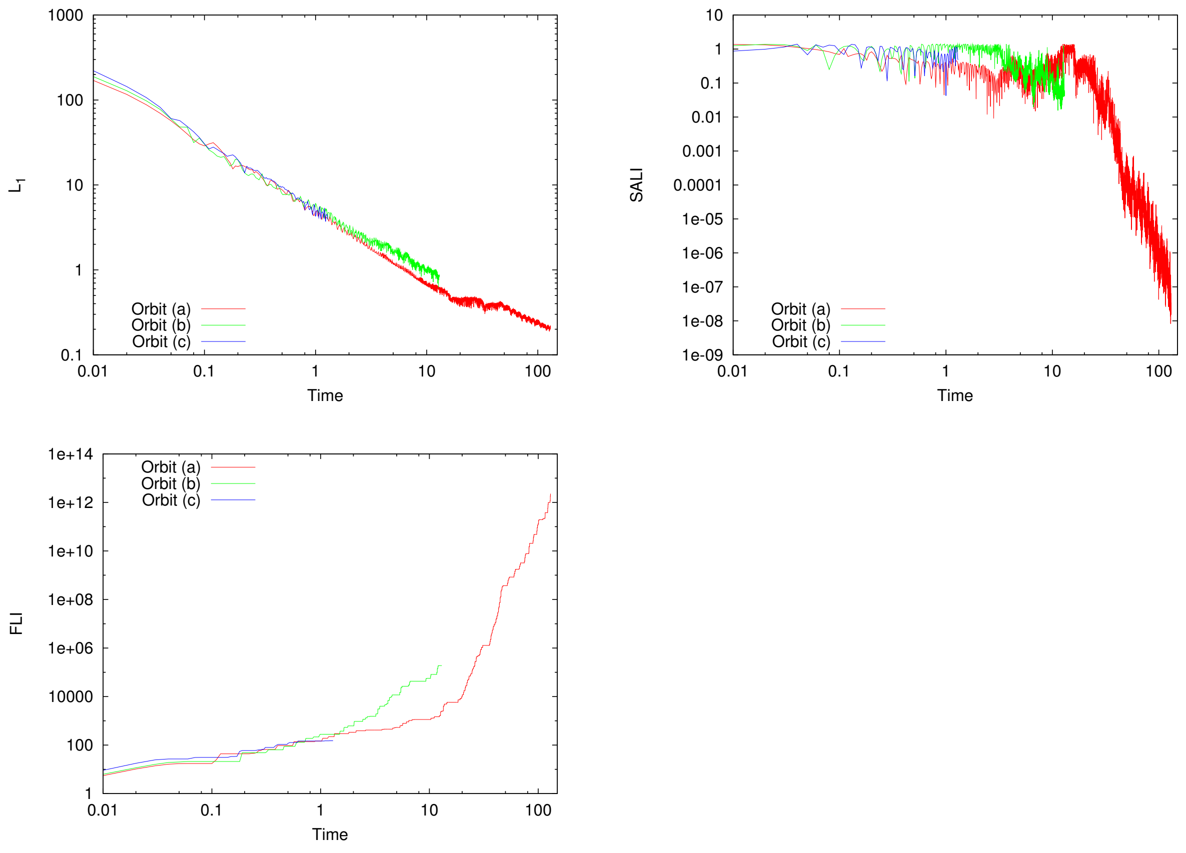

Figure A.3: Examples of the time evolution for the L1 (top left panel), the SALI (top right panel) and the FLI (bottom left panel) for the 3 orbits of the example.